Tarea Validación III

Título

Integrantes

Araiza Verdugo Angel Abraham

Santillán León Fernando Antonio

Prof: Zuriel Dathan Mora Felix

1. Archivo README.md (Documentación General)

TAREA DE VALIDACIÓN III: Algoritmo Genético para el Problema del Viajante (TSP)

1. Descripción del Proyecto

Este proyecto implementa un Algoritmo Genético (AG) diseñado para resolver una instancia del Problema del Viajante (Traveling Salesperson Problem - TSP). El objetivo es encontrar la ruta más corta (de menor distancia) que visita un conjunto predefinido de municipios una sola vez y regresa al punto de inicio (ciclo cerrado).

El AG utiliza una codificación de orden (permutación de municipios) y emplea los siguientes componentes clave:

Función de Aptitud: El inverso de la distancia total** de la ruta (maximizar aptitud = minimizar distancia).

Selección: Combinación de Elitismo y Selección por Ruleta para elegir a los padres.

Cruce (Crossover): Un operador de cruce para permutaciones que garantiza que el hijo es una ruta válida (sin duplicados ni omisiones).

Mutación: Mutación por Intercambio (Swap), que intercambia dos municipios en la ruta.

2. Estructura y Modularización del Código

El código está estructurado en clases y funciones, separando la representación de datos, la evaluación y la lógica del algoritmo.

Componente	Clase/Función	Responsabilidad
Representación	`municipio`	Almacena coordenadas (x, y) y calcula la distancia euclidiana.

Evaluación `Aptitud` Calcula la distancia total del ciclo cerrado y el valor de aptitud (`1 / distancia`).
Inicialización `crear_individuo` Genera una ruta aleatoria (un individuo/cromosoma).
Control del AG `AlgoritmoGenetico` Clase principal que gestiona el flujo evolutivo (`poblacion_inicial`, `ejecutar_algoritmo`, y operadores genéticos).
Operador Genético `reproduccion` Ejecuta el cruce para generar un hijo válido.
Operador Genético `mutacion` Aplica la mutación por intercambio a un individuo.

3. Requisitos y Ejecución

Requisitos de Dependencias

El script requiere **Python 3.x** y las siguientes librerías externas:

`numpy`

`pandas`

Instalación:

```bash
pip install numpy pandas
```

Instrucciones de Ejecución

1. Guarde el código como algoritmo_genetico.py.
2. Ejecute desde su terminal:

Bash

```
python algoritmo_genetico.py
```

El script ejecutará automáticamente los casos de prueba formales para validar las funciones críticas antes de iniciar la simulación del AG.

2. Archivo `TEST_RESULTS.md` (Resultados de Pruebas Formales)

markdown

RESULTADOS DE PRUEBAS FORMALES - ALGORITMO GENÉTICO (TSP)

A continuación, se documentan los resultados de la sección `ejecutar_casos_prueba()` para validar la funcionalidad de los componentes principales del algoritmo.

1. Validación de Aptitud y Distancia (Clase `Aptitud`)

Ruta de Prueba: Ciclo cerrado entre A(0,0), B(3,0), C(0,4).

Cálculo Manual Esperado: Distancia = d(A,B) + d(B,C) + d(C,A) = 3 + 5 + 4 = 12.0. Aptitud = 1 / 12.0 ≈ 0.0833.

Caso de Prueba Métrica Esperado Obtenido Resultado

C1.1 Distancia Total de Ruta 12.00 12.00 PASÓ

C1.2Aptitud Calculada 0.0833 0.0833 PASÓ

2. Validación de Cruce (Método `reproduccion`)

Caso de Prueba Progenitores (Ejemplo) Validación Resultado

C2.1P1=`[A, B, C]`, P2=`[C, A, B]` Se comprueba que el hijo es una ruta de longitud 3 y contiene los municipios {A, B, C}.

3. Validación de Mutación (Método `mutacion`)

Objetivo: Verificar que la mutación por intercambio cambia la ruta Y que la ruta resultante sigue siendo una permutación válida.

Caso de Prueba	Individuo Inicial (Ejemplo)	Validación	Resultado
C3.1	`[A, B, C]`	El individuo mutado es diferente al original Y el set de municipios permanece {A, B, C}.	

Conclusión de la Implementación

Los casos de prueba confirman que los componentes esenciales del Algoritmo Genético, especialmente la función de aptitud y los operadores de cruce y mutación, funcionan correctamente para el problema de la codificación de orden (TSP), asegurando la validez y la integridad de las soluciones generadas.